# EXHIBIT 3

IEICE TRANS. COMMUN., VOL.E88–B, NO.5 MAY 2005



PAPER

# Flow-Level Multipath Load Balancing in MPLS Network

**Zenghua ZHAO**[†a)], **Yantai SHU**[†], **Lianfang ZHANG**[†], ***and*** **Oliver YANG**[††b)], ***Nonmembers***

**SUMMARY** Multi-Protocol Label Switching (MPLS) can efficiently support the explicit routes setup by the use of Label Switched Paths (LSPs) between an ingress Label Switched Router (LSR) and an egress LSR. Hence it is possible to distribute the network traffic among several paths to achieve load balancing, thus improving the network utilization, and minimizing the congestion. The packet-level traffic characteristics in the Internet is so complex that it is natural to do traffic engineering (TE) and control at the flow level. The emerging Multi-Protocol Label Switching (MPLS) has introduced an attractive solution to TE in IP networks. The main objective of this paper is to balance traffic at the flow level among the parallel Label Switched Paths (LSPs) in MPLS networks. We introduce a multipath load-balancing model at the flow level. In this model, each LSP is modeled as an M/G/1 processor-sharing queue. The load-balancing problem is then considered as an optimization problem. Based on the analysis of the model, we propose a heuristic but efficient mechanism that can make good use of the traffic characteristics at the flow level. Packet disorder is avoided effectively by dispatching packets belonging to one flow to the same path. This mechanism only need to be implemented in the ingress LSRs and the egress LSRs, while the intermediate LSRs only forward the packets. Apart from discussing the traffic allocation granularity, and the implementation issues in details, we have also performed extensive simulations using NS-2 with MPLS modules. The simulation results show that the load through the network is well balanced so that the network throughput is improved and the delay is decreased efficiently.

***key words:*** *flow level model, multipath load balancing, MPLS network*

## 1. Introduction

The purpose of traffic engineering (TE) is to improve network performance through the optimization of network resources [1]. The emerging Multi-Protocol Label Switching (MPLS) technology has introduced an attractive solution to TE in IP networks. MPLS can efficiently support the explicit routes setup through the use of Label Switched Paths (LSPs) between the ingress Label Switched Router (LSR) and the egress LSR [2]. Hence it is possible to balance the traffic through the network, thus improving the network utilization and minimizing the congestion. Several researchers have proposed some solutions to balance the load in MPLS networks.

An analytical framework is presented in [3] where different models with different objective functions for the best-effort and the expedited-forwarding traffic are established. However, it is too complex to implement in operational networks. Moreover, it is impossible to obtain the delay contributed by the cross traffic which is required to calculate how much traffic should be shifted among the LSPs. Another load balancing mechanism called MATE (MPLS adaptive traffic engineering) [4] develops its algorithm based on the gradient-project method, and hence unavoidably inherits its disadvantages, namely its high complexity and slow convergence. The ECMP (Equal Cost Multi-Path) algorithm [5] attempts to distribute the traffic as equally as possible among the equal-cost paths. Since the link costs are static and bandwidth constraints are not considered, the traffic allocation is independent of the congestion status of each path. Therefore, it is possible that one of the paths will be more congested than the other.

Traffic distribution is an important part of the implementation of a load-balancing algorithm. The two pieces of work [3], [4] above have adopted the hash function method. In this method, traffic is first distributed into $N$ bins according to a hash function, where the number of bins determines the minimum amount of the traffic that can be shifted. If the total incoming traffic is of rate $R$ bps, each bin would receive an amount of $r = R/N$ bps. The traffic from the $N$ bins is then mapped into several LSPs according to their load-balancing algorithms. Using the hash function on the IP fields, the incoming packets with same destination can be distributed into the same bin. It seems that it can reduce the possibilities of having packets arrive at the destination out of order as described in [4]. But unfortunately, it does not work well. In order to achieve load balancing, the traffic should be shifted dynamically among the multiple LSPs in the unit of one bin (i.e. $1/N$ of the total traffic). Therefore, the traffic from one bin will be shifted from one LSP to another during the load balance phase. In other words, the packets belonging to one flow may end up going along different LSPs, and therefore it is difficult to avoid packet disordering that can impact on the performance of TCP.

It is in general difficult to model the performance of the Internet traffic with a self-similar characteristic at the packet level [6]. Therefore, for traffic control purpose, it is natural to characterize traffic at the flow level instead. Flows may be broadly categorized as streaming or elastic according to their nature [7]. It has been pointed out [8] that the arrival of flows in the Internet could be modeled as a Poisson process. Therefore, a new service model at the flow level would shed

Manuscript received May 6, 2004.
Manuscript revised September 6, 2004.

[†]The authors are with the Faculty of Computer Science, School of Electronics and Information Engineering, Tianjin University, Tianjin, 300072, China.
[††]The author is with the Faculty CCNR Lab., School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada.
a) E-mail: zenghua@tju.edu.cn
b) E-mail: yang@site.uottawa.ca

DOI: 10.1093/ietcom/e88–b.5.2015

some light on network engineering issues, including flow-aware routing, pricing, admission control, and bandwidth sharing.

When many flows (fluid) share a single link evenly, an M/G/1 processor sharing (PS) queuing model [8] can be used. Also, since a symmetric queue is quasi reversible for a general service time distribution [9], we can exploit these properties to be explained in Sect. 2. Finally, an M/G/1 PS queue can approximate the network of transmission links more realistically [11], [chap 3] than models using the kleinrock's Independence Assumption [10].

In view of the above, we have adopted a PS (processor sharing) queueing network to model the MPLS networks according to traffic characteristics at the flow level. The main objective of this paper is to balance the traffic at the flow level among multiple parallel LSPs in an MPLS network. A per-flow traffic distribution that will avoid packet disorder as much as possible is employed. Based on the analytical model, we propose a heuristic but efficient algorithm. Unlike the ECMP, we propose to measure the link status (such as delay) dynamically according to where the traffic is distributed. Our simulation results demonstrate that our algorithm can distribute the traffic to different LSPs and balance their loads approximately.

The rest of the paper is organized as follows. Section 2 gives the flow-level analytical model. Section 3 proposes a heuristic algorithm to balance the load in the MPLS networks based on the analytical model. Section 4 describes the implementation issues of the load-balancing mechanism. In Sect. 5, we evaluated and discuss the performance of the mechanism. Section 6 concludes this paper.

## 2. The Analytical Model at the Flow Level in MPLS Networks

In this section, we first introduce the traffic characteristics at the flow level before we give our multipath load-balancing model in MPLS networks. Using a simple case study, we want to shed some light on the heuristic load-balancing algorithm proposed in the next section.

### 2.1 Flow-Level Traffic Characteristics

We define a flow to be a sequence of packets having the same identifier (source address, destination address, port number, etc.). Of the two broad categories of flows, we only consider the elastic type, since elastic traffic nowadays makes up the majority of the Internet traffic. Flow arrivals are assumed to be Poisson and the flow size has a Pareto distribution given by

$$P[\text{size} \le x] = 1 - x_0/x^{\beta}, \quad \text{for} \quad x \ge x_0 > 0, \quad \text{with} \quad 1 < \beta \le 2. \tag{1}$$

Note that this distribution has a finite mean and infinite variance. Like [8], we model a single bottleneck link fairly shared by elastic flows as an M/G/1 processor sharing a (PS) queue. Let $\rho$ be the link utilization. Then for $\rho < 1$, the number of flows $X(t)$ in progress has a geometric distribution,

$$P[X(t) = n] = \rho^n(1-\rho).$$

### 2.2 The Model

We consider an MPLS network where multiple parallel LSPs exist between any given ingress LSR and egress LSR pair. The main objective is to distribute the traffic at each ingress LSR among the multiple LSPs so as to balance the load through the network and thus improving the network performance. Given an objective function, we can view this as an optimization problem [3], [4], [11].

Now the MPLS network considered could be approximated as a PS queueing network, where each PS queue corresponds to a link, and an arrival flow as a customer. According to the basic theorem of quasi-reversible queueing networks [9], the stationary number of flows on the links is independent and their invariant distribution has a product-form. For simplicity, we assume further that all links are identical and the bandwidth of each link is equal to 1. Let $L$ denote the link set, and $R$ the set of LSPs. Let $\nu$ be the total flow arrival rate entering the network, $\nu_r$ the flow arrival rate on LSP $r$, and $\mu_r^{-1}$ be the average flow size. Then $\rho_r = \nu_r/\mu_r, \sum \nu_r = \nu$, then according to the stationary product-form distribution of customer number on the quasi-reversible queueing network [9][Theorem 3.3.2], $P(n_1, \ldots, n_L) = \prod_{l \in L} \rho_l^{n_l}(1-\rho_l)$ and the average flows numbers on link $l$ can be obtained via

$$E[n_l] = \frac{\rho_l}{1-\rho_l}, \quad \text{for all} \quad l \in L.$$

Based on the above discussion on network operation and assumptions, we can now pose our load-balancing problem as follows.

$$\begin{aligned} &\text{Minimize} \sum_{l \in L} \frac{\rho_l}{1-\rho_l}, \quad \rho_l < 1, \\ &\text{subject to} \sum_{r \in R} \nu_r = \nu. \end{aligned} \tag{2}$$

This is an optimization problem, and the solution could be given by a gradient-projection algorithm. But such an algorithm is complex and not robust. For example, it is difficult to determine the convergence step size of the algorithm. To gain more insights into our model, we shall first conduct a simple network case study with only one ingress-egress nodes pair with two LSPs between them. We shall analyze it in details in the section below. Our experience will allow us to extend our model to several LSPs (and even to the whole MPLS network), and give us guidance in deducing a heuristic equation for traffic distribution in Sect. 3.

#### 2.2.1 A Simple Case Study

In Fig. 1, we consider only one ingress-egress LSR pair with

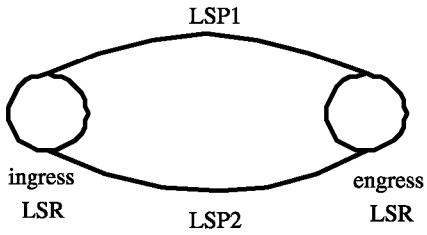



**Fig. 1** A case of flow level multipath load-balancing model traffic.

two LSPs between them in an MPLS network. There are two types of traffic: engineered traffic and cross traffic. Engineered traffic is the traffic that needs to be balanced, and cross traffic is the background traffic that we have no control over. Engineered traffic flows traverse the network from the ingress LSR to the egress LSR.

We assume that the total engineered traffic flow rate is $\nu$. Using the splitting mechanism discussed below, engineered traffic is distributed into the two LSPs, each with $\nu_1$ and $\nu_2$ respectively. We let $\tilde{\nu_1}$ and $\tilde{\nu_2}$ be the cross traffic on LSP1 and LSP2. With an average flow size of $\mu^{-1}$, our optimal load balancing should therefore minimize

$$\frac{\rho_1 + \tilde{\rho}_1}{1 - \rho_1 - \tilde{\rho}_1} + \frac{\rho_2 + \tilde{\rho}_2}{1 - \rho_2 - \tilde{\rho}_2}, \tag{3}$$

subject to $\nu_1 + \nu_2 = \nu$.

where $\rho_i = \nu_i/\mu$, and $\tilde{\rho}_i = \tilde{\nu}_i/\mu, (i = 1, 2)$. Using the Lagrangian Multiplier method, it is easy to obtain the minimized condition of

$$\frac{1}{(1 - \rho_1 - \tilde{\rho_1})^2} = \frac{1}{(1 - \rho_2 - \tilde{\rho_2})^2}$$

After further substitution of $\rho_i = \nu_i/\mu, (i = 1, 2)$ to the above equation, we can obtain:

$$\frac{1}{\mu - \nu_1 - \tilde{\nu}_1} = \frac{1}{\mu - \nu_2 - \tilde{\nu}_2}. \tag{4}$$

Since the average response time $T$ of M/G/1 PS system is:

$$T = \int \frac{x}{1 - \rho} dG(x) = \frac{\frac{1}{\mu}}{1 - \rho}$$

where $G(x)$ is the general flow size distribution. Therefore, the average system response time before the engineered traffic arrives is

$$T_i = \frac{\frac{1}{\mu}}{1 - \tilde{\rho}_i} = \frac{1}{\mu - \tilde{\nu}_i}, (i = 1, 2). \tag{5}$$

By substituting (5) in (4), and rewriting it, we can obtain the equilibrium point condition:

$$\nu_1 - \nu_2 = \frac{1}{T_1} - \frac{1}{T_2}. \tag{6}$$

In order to further make the following algorithm robust, we have relaxed the equilibrium condition to

$$\Delta\nu_{12} = \nu_1 - \nu_2 \propto \left(\frac{1}{T_1} - \frac{1}{T_2}\right). \tag{7}$$

After studying the above, we can now consider the general $N$ paths case where the minimization problem now becomes

$$\text{Minimize} \sum_{i=1}^{N} \frac{\rho_i}{1 - \rho_i},$$

subject to $\nu_1 + \cdots + \nu_N = \nu$,

This again can be solved by the Lagrangian method, and the equations similar to the above can be obtained. That is,

$$\frac{1}{\mu - \nu_i - \tilde{\nu}_i} = \frac{1}{\mu - \nu_j - \tilde{\nu}_j},$$

for all $i, j = 1, \ldots, N$.

In the same way, we can obtain the following relaxed conditions:

$$\Delta\nu_{ij} = \nu_i - \nu_j \propto \left(\frac{1}{T_i} - \frac{1}{T_j}\right), \quad \forall i, j = 1, 2, \ldots, N, i \neq j. \tag{8}$$

## 3. The Heuristics

From the above discussion, we can deduce a method to distribute traffic among LSPs sharing the same ingress-egress LSR pair. Equation (8) shows that the difference of traffic distributed on any two LSPs should be proportional to the difference of the reciprocal of the average packet delays of the corresponding LSPs. However, in order to obtain a sensible deduction with a simple form, we only take the queueing delay into considerations in the theoretical analysis. Unfortunately, the propagation delay of LSPs cannot be neglected, such as an MPLS core network. When loading traffic, we prefer to load more traffic on the LSP with lower queueing delay and propagation delay. In addition, the above information obtained is static and it is difficult to use it directly in real networks. In order to make the distribution method discussed above more feasible, we propose a heuristic load balancing equation:

$$w_i = \frac{1/d_i}{\sum_k 1/d_k}, \tag{9}$$

where $w_i$ is the weight of LSP$i$, determining how much traffic should be distributed to LSP$i$, and $d_i$ is the total delay (including propagation delay) of LSP$i$ which can be approximated by the round-trip time (RTT) measured dynamically. The difference between the weights of LSP$i$ and LSP$j$ is

$$\Delta w_{ij} = w_i - w_j = \left(\sum_k 1/d_k\right)\left(\frac{1}{d_i} - \frac{1}{d_j}\right), \quad \text{i.e. } \Delta w_{ij} \propto \left(\frac{1}{d_i} - \frac{1}{d_j}\right). \tag{10}$$

Since (10) is in accordance with (8), (9) is a feasible solution.

## 4. Implementation Issues

In this section, we will discuss further the implementation issues involved in our mechanism.

### 4.1 Flow-Level Load-Balancing Mechanism in MPLS Networks

Our mechanism is implemented only in the ingress LSRs and egress LSRs in MPLS networks. The ingress LSRs distribute the traffic input to the network at the flow levels according to our algorithm. The intermediate LSRs do nothing but forwarding the traffic through them. This adheres to the philosophy of the design of a backbone network: keep complexity at the edge while making the middle as simple as possible. Moreover, the mechanism is asynchronous, and it is unnecessary for the ingress LSRs to exchange information.

Our mechanism functions implemented in the ingress LSR are shown in Fig. 2. When a packet arrives, the **flow distribution module** in the ingress LSR first decides if it belongs to a new flow. If so, the flow will be assigned to a LSP according to our algorithm discussed above. Otherwise, the ingress LSR will search the **flow-state table**, and distribute the packet to the LSP the existing flow the packet belongs to. The **flow-state table** is maintained by the **flow distribution module**, including the flow state information described below. The **average delay estimation** is used to measure the average delay of LSPs between the ingress LSR and the egress LSR.

### 4.2 Traffic Allocation Granularity

For the sake of simplicity, we only consider the elastic flows that occupy most of the traffic on the Internet. The allocation granularity is larger at the flow level than that at the packet level. Although Krishnan et al. [12] advocated per-packet allocation granularity, we would argue that per-flow allocation granularity is fine enough for traffic distribution. The majority of elastic flows are very short (they are less than 1 kbytes), and only a few of them are long-lived flows [13]. Therefore, for any given ingress LSR and egress LSR pair, the number of flows between the two LSRs is potentially much higher than the number of the parallel LSPs connecting them. This is similar to what was suggested by Lai [14]. Moreover, the direct advantage of per-flow traffic distribution is that it decreases the disorder phenomenon whereas the per-packet mechanism does not. This is due to the fact that, in the per-flow traffic distribution, all packets in a flow will go along the same path as long as the path is determined.

### 4.3 Flow State Table

In order to implement per-flow traffic distribution, we maintain a **flow-state table** in each ingress LSR as shown in Table 1. Each entry in this table has 4 fields: 1) The *flow identification* field to record the 5-tuples of a flow (source address, destination address, source port, destination port and protocol ID). 2) An *Index* field to speed up the searching process. It is a short word and can be obtained by transforming *flow identification* through a particular function. 3) The *LSP#* field to indicate the number of LSP a flow is going through. 4) The *Last_packet_arrival_time* field to record the arrival time of the last packet of a flow.

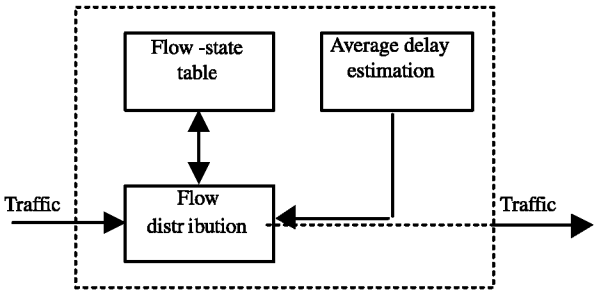



**Fig. 2** The mechanism in an ingress LSR.

**Table 1** Flow-state table maintained by ingress LSR.

| *index* | *flow identification* | *LSP#* | *last_packet_ arrival_time* |
|---|---|---|---|

Note that it is unnecessary to maintain the **flow-state table** in the core LSRs. In addition, the size of the table is a linear function of the number of the current active flows through the ingress LSR. In fact, the *flow identification* can be removed without any impact on the process in order to save memory. On the other hand, the speed of the CPU nowadays is getting faster and faster, and the memory cheaper and cheaper, therefore, we can improve the performance of the MPLS network at the cost of maintaining the **flow-state table**.

### 4.4 Flow Distribution

When a packet arrives, the ingress LSR first searches its **flow-state table** to see if this packet belongs to an existing flow. If so, the ingress LSR will distribute the packet to an LSP with the same *LSP#* from the entry, and update the *Last_packet_arrival_time* field with the arrival time of the current packet. Otherwise, a new flow must have started, and the ingress LSR must prepare a new entry and insert it into the **flow-state table**. Specifically, the LSR records the *flow identification*, compute the *index* through transformation, record the current time (i.e. the packet arrival time) in the *last_packet_arrival_time* field, and determine the *LSP#* according to our algorithm.

The ingress LSR will check the *last_packet_arrival_time* of each flow in the **flow-state table** when dealing with a new flow. If the difference from the current time is more than the threshold $T$, then it will delete the entry of the corresponding flow from the **flow-state table**.

### 4.5 Delay Estimation

In order to monitor the real-time information of each LSP,

we have adopted a probing mechanism. We send probing packets periodically to each LSP, and measure their round-trip time (RTT). We then estimate the path delay using an exponential weighted moving average shown below [15].

$$ED_i = (1 - \alpha) \times ED_{i-1} + \alpha \times SRTT_i$$

where $ED_i$ is the $i$th estimated delay, $SRTT_i$ is the $i$th measured RTT.

The **flow distribution module** will use the estimated delay to distribute the traffic according to our algorithm discussed in the above section.

## 5. Performance Evaluation

We use NS-2 [16] with an MPLS module [17] to implement our simulation. To illustrate the feasibility and generality of our method, two simulation scenarios are established. The first scenario is the simplest case with two paths between one IE (Ingress LSR - Egress LSR) pair. The second scenario considers four IE pair where the intermediate links are shared by the LSPs from different IE pairs. In the simulation, a flow is simply a TCP connection. Flow arrivals are Poisson with a mean of 20 flows per second. The flow size has a Pareto distribution with $x_0 = 4, \beta = 1.5$ (cf. (1)).

### 5.1 Simulation Scenario 1

The network topology is shown in Fig. 3 where each link has a bandwidth of 60 Mbps and a delay of 10ms. There is only one ingress LSR and egress LSR pair with two parallel LSPs (numbered as LSP1 and LSP2) between them. Flows enter the MPLS network from the ingress LSR and exit from the egress LSR. In order to verify the efficiency of our mechanism, we have introduced cross traffic to change the link bandwidth available to flows entering the network. The cross traffic uses the on-off mode CBR (Constant Bit Rate) traffic, and goes through LSR1 and LSR3.

#### 5.1.1 Performance Evaluation

We first obtain the time evolution of offered load distribution on LSP1 and LSP2 without cross traffic in the network. The results are shown in Fig. 4 which we shall use a reference for later comparison.

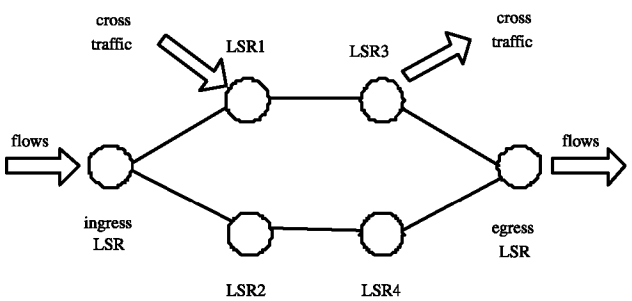



**Fig. 3** Network topology 1.

We then start the cross traffic running between time $t$ = 50 s and $t$ = 150 s at a data rate of 30 Mbps in order to reduce the available bandwidth on LSP1 during this period. As can be seen from Fig. 5 the offered load is adaptively shifted from LSP1 to LSP2, and the load is balanced between the two LSPs very well. Figure 6 and Fig. 7 illustrate the end-to-end delay and throughput of each flow respectively. Note that we have rearranged the flow sequence in Fig. 7(b) according to the throughput in order to show the results clearly. The results demonstrate clearly that the end-to-end throughput has been increased while the end-to-end delay has been decreased, thus showing the network performance has been improved significantly when compared with that in the original MPLS network.

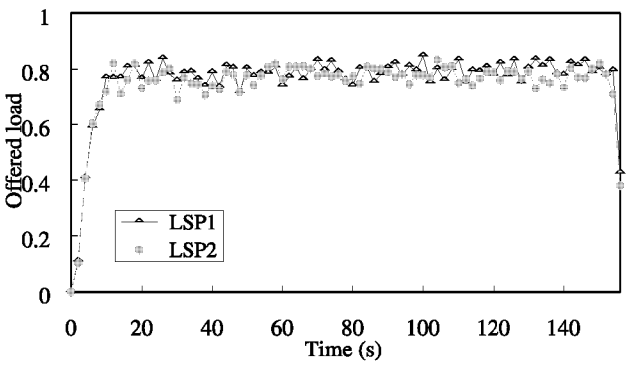



**Fig. 4** Offered load distributed on LSP1 and LSP2 without cross traffic.

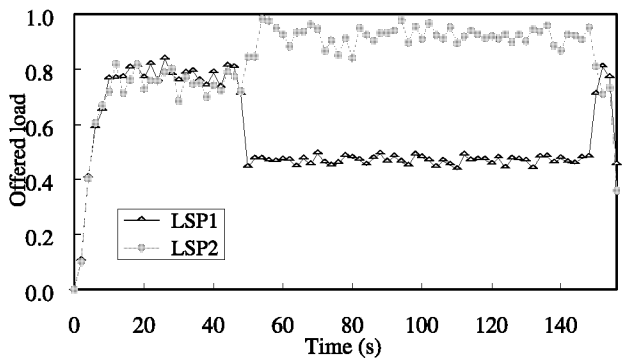



**Fig. 5** Offered load distributed on LSP1 and LSP2 with cross traffic.

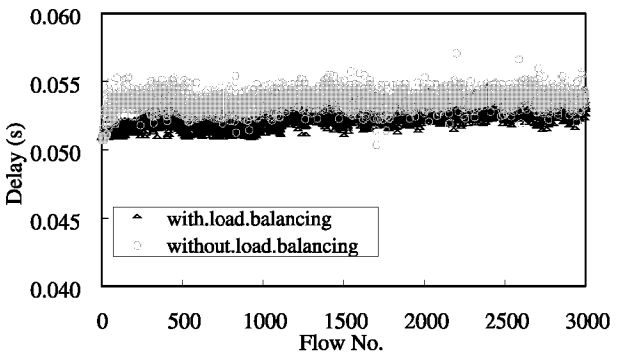
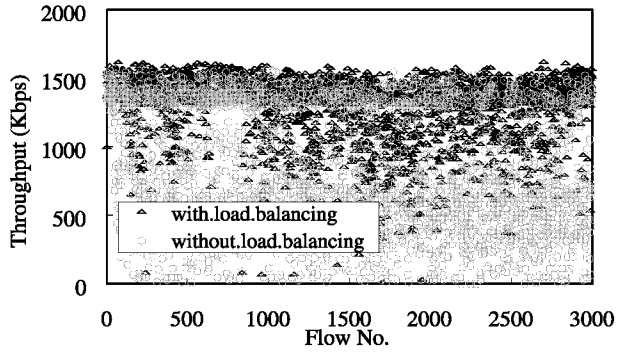
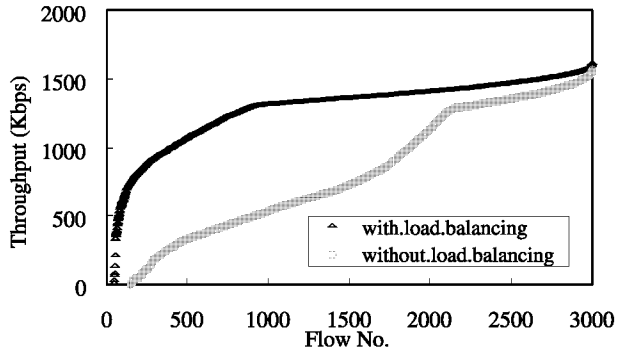



**Fig. 6** End-to-end delay of flows.




(a) Flow sequence number same as generated.




(b) Flow sequence number rearranged according to throughput.

**Fig. 7** End-to-end throughput of flows.

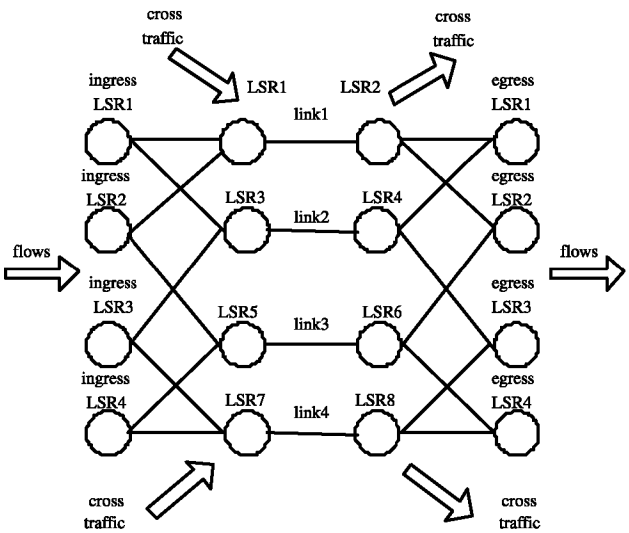



**Fig. 8** Network topology 2.

## 5.2 Simulation Scenario 2

We have used in the last section a very simple network topology to support the theoretical load balancing equation derived earlier. In this section, we shall design a more complicated but practical MPLS network topology as shown in Fig. 8. There are four IE pairs with two parallel LSPs between each of them. The LSPs from different IE pairs share some intermediate links. For example, link1 is shared

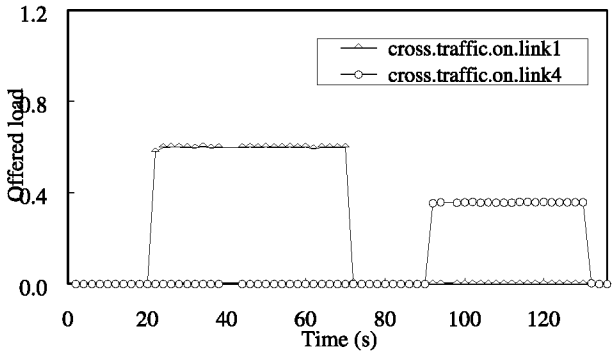



**Fig. 9** The cross traffic added to the network.

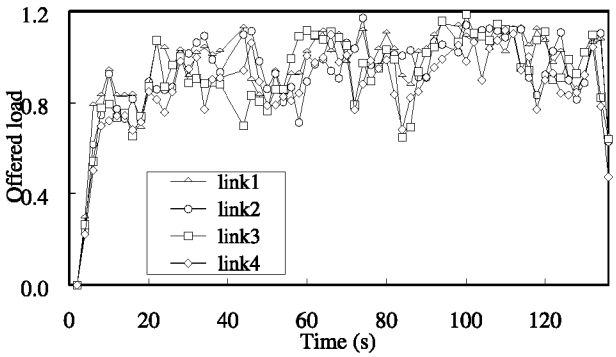



**Fig. 10** Offered load distributed on links without cross traffic.

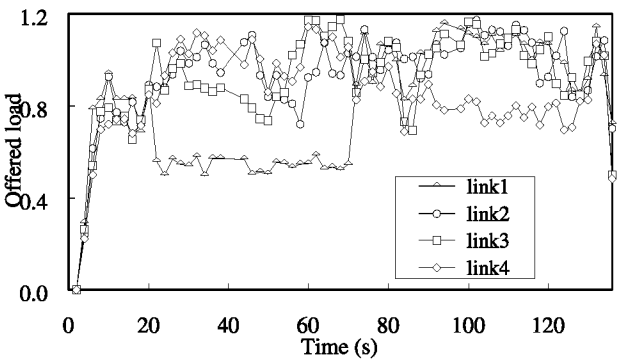



**Fig. 11** Offered load distributed on links with cross traffic.

by the LSP(Ingress-LSR1→LSR1→LSR2→Egress-LSR1) and another LSP(Ingress-LSR2→LSR1→LSR2→ Egress-LSR2). The cross traffic with different rates and durations are added to link1 and link4 at different time as shown in Fig. 9.

### 5.2.1 Performance Evaluation

We have collected the time evolution of the offered load on the four links. Figure 10 shows the traffic has been allocated evenly to the four links when there is no cross traffic. Figure 11 shows how the traffic on link1 are shifted partly to other links during the interval between 20 s and 70 s when the cross traffic occupying about 50% bandwidth

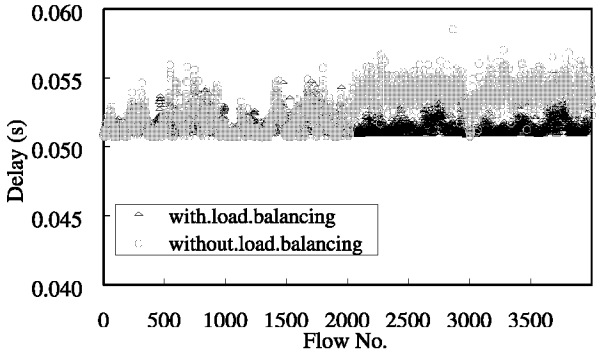
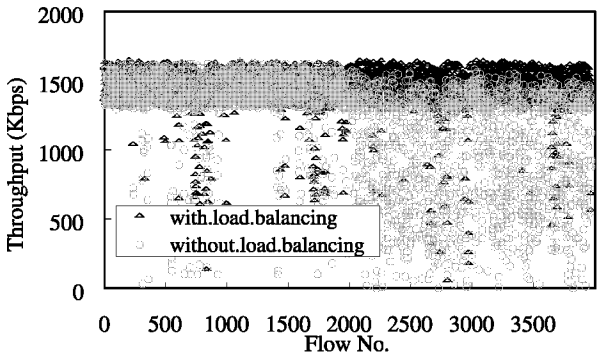
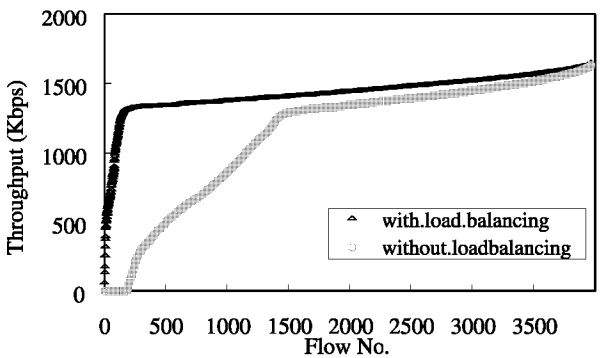



**Fig. 12** End-to-end delay of flows.




(a) Flow sequence number same as generated.




(b) Flow sequence number rearranged according to throughput.

**Fig. 13** End-to-end throughput of flows.

is added to link1. Similar observations can be made during 90 s and 130 s. It's worthwhile to note that the traffic has been shifted adaptively according to the strength of the cross traffic(in other words, according to the available bandwidth). The end-to-end delay and throughput of each flow are shown in Fig. 12 and Fig. 13 respectively. Again, when compared with the original MPLS network, our method improves flows performance. That is, the end-to-end throughput is increased and the end-to-end delay is decreased.

In summary, our method can balance the load very well.

## 6. Conclusions

In this paper, we have analyzed a flow-level model in order to balance the load among parallel LSPs in the MPLS networks. The purpose is to improve network performance and to minimize the congestion. Our analysis allows us to propose a flow-level mechanism. Although a heuristic, our mechanism is efficient because it employs per-flow traffic distribution to avoid the packet disorder. It is also easy to implement in the ingress LSRs and egress LSRs. Our simulation results show that the load through the network is so well balanced that the network throughput can be improved and the delay decreased significantly.

## Acknowledgement

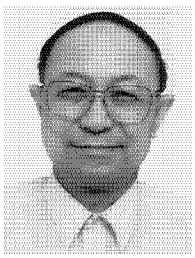
This research is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 60472078 and No. 90104015, by the Cisco CCRP under Grant No. 20029303, by the NSF of Tianjin under Grant No. 023601111, and No. 043600311, and by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant No. RGPIN-2003-42878. Thanks to Hongmei Wang for her work on simulation.

### References

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," Internet RFC 2702, Sept. 1999.
[2] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet RFC 3031, Jan. 2001.
[3] E. Dinan, D. Awduche, and B. Jabbari, "Analytical framework for dynamic traffic partitioning in MPLS networks," IEEE ICC'00, pp.1604–1608, New Orleans, June 2000.
[4] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "Mate: MPLS adaptive traffic engineering," Proc. IEEE INFOCOM'01, pp.22–26, Anchorage, Alaska, April 2001.
[5] D. Awduche, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet traffic engineering," RFC 3272, May 2002.
[6] A. Feldmann, P. Huang, A.C. Gilbert, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," Computer Communication Review, Sigcomm 29, no.4, 1999.
[7] J.W. Roberts, "Traffic theory and the Internet," IEEE Commun. Mag., pp.94–99, Jan. 2001.
[8] S.B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J.W. Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," Proc. ACM SIGCOMM '01, pp.27–31, San Diego, California, U.S.A. Aug. 2001.
[9] Kelly, Reversibility and Stochastic Networks, Wiley, Chichester, 1979.
[10] L. Kleinrock, Queueing Systems, vol.2. Wiley, New York, 1976.
[11] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, 1992.
[12] R. Krishnan and J. Silvester, "Choice of allocation granularity in multipath source routing schemes," Proc. IEEE Conference on Computer Communications, pp.322–329, San Francisco, CA, March 1993.
[13] M.F. Arilitt and C. Williamson, "Web server workload characterization: The search for invariants," Proc. ACM SIGMETRICS'96 Conference, pp.126–137, Philadelphia, April 1996.
[14] W.S. Lai, "Bifurcated routing in computer networks," Computer Communication Review, vol.15, no.3, pp.28–49, 1985.
[15] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," Computer Communication Review, vol.17, no.5, pp.2–7, Aug. 1987.

[16] NS-2 Network Simulator, http://www.isi.edu/nsnam/ns/
[17] G. Ahn and W. Chun, "Overview of MPLS networks simulator: Design and Implementation," Chungnam National University, Korea.

**Zenghua Zhao** is an Assistant Professor of Computer Science at Tianjin University, China. She received her M.S. in Computer Science, Tianjin University, China, in 2000. Her current interests are focussed on computer networks, modeling and simulation, Wireless Multimedia network.



**Yantai Shu** is a professor of computer science at Tianjin University, China, serving as vice president of the university from 1993–1997. His current interests are focused on computer communication networks, wireless networks, real-time systems, modeling and simulation. He received Chinese equivalent to B.S., M.S., and Ph.D. in electronics engineering from Tianjin University. From 1974 until 1991 he was employed by the Institute of Plasma Physics, Academia Sinica in research positions. He is a member of the IEEE and the ACM. He has published more than 120 papers and contributed to one book.

**Lianfang Zhang** is a Professor in Computer Science at Tianjin University, China. He received his M.S. in Institute of Mathematics, Academia Sinica in 1981. His research interests are in performance evaluation of computer networks.

**Oliver Yang** is a Professor in the School of Information Technology and Engineering at University of Ottawa, Ontario, Canada. Dr. Yang received his Ph.D. degree in Electrical Engineering from the University of Waterloo, Ont., Canada in 1988. His research interests are in the modeling, analysis and performance evaluation of computer communication networks, their protocols, services and interconnection architectures, queueing theory, simulations, computational algorithms and their applications. Dr. Yang has published more than 200 technical papers in wireline, wireless and photonic networks. He is currently an editor of the IEEE Communication Magazine.